Università di Verona, Facoltà di Scienze MM.FF.NN.

Insegnamento di Architetture Multimediali

Strumenti e protocolli di rete all'opera

Davide Quaglia, Francesco Stefanni, Giuseppe Di Guglielmo

Scopo di questa esercitazione è imparare l'utilizzo di tool software per:
1) impostare ed ottenere informazioni sulla configurazione dei vari PC presenti su una rete;
2) analizzare il traffico di una rete osservando il comportamento dei vari protocolli;
3) analizzare il comportamento di alcune applicazioni di rete come il Web e la posta elettronica;
4) analizzare la capacità disponibile sulla rete.

1 Analisi ed impostazione della configurazione di rete

Per visualizzare le impostazioni di rete del proprio PC con Linux occorre dare il comando

```
$ /sbin/ifconfig -a
```

che visualizza la lista delle interfacce di rete e, per ognuna, le impostazioni MAC e IP. L'interfaccia ethernet viene indicata con `eth0` e si può osservare l'indirizzo MAC (6 byte esadecimali), l'indirizzo IP (nel formato *dotted*) e la netmask (nel formato *dotted*).

Per visualizzare l'indirizzo del default gateway occorre usare il comando

```
$ /sbin/route
```

che stampa a video una routing table che, nel caso di un PC con una sola interfaccia IP, è ridotta a due sole righe:
- quella delle destinazioni raggiungibili direttamente (stessa rete)
- tutte le altre, raggiungibili tramite appunto il default gateway.

Si possono usare gli stessi programmi per configurare l'indirizzo IP e la netmask in relazione ad una interfaccia di rete, e il default gateway:

```
$ /sbin/ifconfig eth0 157.27.241.11X netmask 255.255.255.0 up
$ /sbin/route add default gw 157.27.241.1
```

NOTA: nel laboratorio gli indirizzi IP seguono questa logica: *157.27.241.110 + numero_macchina*

1.1 Internet Control Message Protocol (ICMP)

Il protocollo Internet Control Message Protocol (ICMP) è stato progettato per riportare anomalie che accadono nel routing di pacchetti IP e verificare lo stato della rete. ICMP è specificato nel RFC 792. La tabella seguente riporta i tipi di pacchetti ICMP. I messaggi ICMP sono imbustati nel pacchetto IP il cui campo Protocol riporta il codice 0x01. I messaggi ICMP hanno un campo *Type* che ne indica il tipo.

I messaggi che riportano anomalie sono, ad esempio, *destination unreachable*, *time exceeded for a datagram* e *parameter problem on a datagram*. I messaggi di verifica della raggiungibilità di un nodo sono *echo request* e *echo reply* . Il messaggio *redirect* indica una condizione di stimolo ad un routing

migliore, in quanto un router è stato attraversato inutilmente (ha dovuto ritrasmettere il messaggio sulla stessa rete da cui lo ha ricevuto). Quando un host riceve un pacchetto di *routing redirect* tratta l'informazione in esso contenuta in modo simile a quella specificata da un comando di `route add` ed associa quindi un router diverso da quello di default a quella destinazione.
Gli ultimi messaggi ad essere stati introdotti nel protocollo ICMP sono *address mask request* e *address mask reply*, per permettere ad una interfaccia di scoprire automaticamente la netmask usata in quella rete.

Valore riportato nel campo Type	Tipo di messaggio
0	Echo Reply
3	Destination Unreachable
4	Source Quence
5	Redirect
8	Echo Request
11	Time Exceeded for a Datagram
12	Parameter Problem on a Datagram
13	Timestamp Request
14	Timestamp Reply
15	Information Request
16	Information Reply
17	Address Mask Request
18	Address Mask Reply

Il tool *ping* utilizza ICMP per verificare la raggiungibilità di un'interfaccia IP (di un host o di un router). Il programma viene lanciato da shell con il comando:

```
$ ping indirizzo_IP
```

Il programma invia un messaggio ICMP Echo request all'indirizzo IP specificato che dovrebbe rispondere con un messaggio ICMP Echo reply.

Esempio:

```
$ ping 157.27.241.1
```

Il programma invia più messaggi e calcola anche il round trip time a seguito della risposta.

NOTA: purtroppo presso gli amministratori di rete si sta diffondendo la consuetudine di bloccare i messaggi ICMP appena raggiungono il primo router della propria rete.

1.2 Address Resolution Protocol (ARP)

Il protocollo ARP permette di conoscere l'indirizzo MAC associato ad un indirizzo IP sulla stessa rete

di livello 2. Le associazioni sono mantenute in una cache temporanea da cui vengono cancellate dopo un certo tempo.
Per visualizzare il contenuto della cache ARP del proprio PC occorre usare il comando

```
$ arp -a
```

1.3 Domain Name Service (DNS)

Domain Name Service (spesso indicato con DNS) è un servizio utilizzato per la risoluzione di nomi di host in indirizzi IP. Il servizio permette così di utilizzare i nomi e le parole di uso comune per fare riferimento ad host su cui risiedono delle applicazioni, ad esempio un sito internet. In pratica un nome di un host viene associato al suo indirizzo IP. Ad esempio il sito di Novell www.novell.com in realtà è solo un modo facile per identificare l'host residente all'indirizzo 130.57.5.25. La possibilità di attribuire nomi simbolici agli indirizzi IP degli host è essenziale per l'usabilità di Internet, perché gli esseri umani trovano più facile ricordare nomi testuali, mentre gli host ed i router sono raggiungibili solo utilizzando gli indirizzi IP numerici.
Il servizio di associazione nomi/IP è realizzato tramite un database distribuito, costituito dai server DNS.
Un nome è costituito da una serie di stringhe separate da punti, ad esempio `it.wikipedia.org`. La stringa più a destra è detta *dominio di primo livello* (o TLD, Top Level Domain), per esempio .org o .it. Un dominio di secondo livello consiste in due parti, per esempio `wikipedia.org`, e così via. Ogni ulteriore stringa a sinistra specifica un'ulteriore suddivisione. Quando un dominio di secondo livello viene registrato all'assegnatario, questo è autorizzato a usare i nomi di dominio relativi ai successivi livelli come `it.wikipedia.org` (dominio di terzo livello) e altri come `some.other.stuff.wikipedia.org` (dominio di quinto livello) e così via; questo meccanismo garantisce l'unicità dei nomi.
Ogni volta che si fa accesso ad un host su Internet specificandone il nome (ad es. un sito web o un server di posta) il nostro PC chiede ad un server DNS l'indirizzo IP corrispondente che sarà poi usato per spedirgli i pacchetti.

La lista con il o i server DNS da contattare deve essere impostata dall'amministratore di rete per ogni host. Su Linux tale lista è contenuta nel file /etc/resolv.conf che può essere visualizzato nel modo seguente:

```
$ cat /etc/resolv.conf
```

In laboratorio tale file dovrebbe essere impostato come segue:

```
$ vi /etc/resolv.conf

search sci.univr.it
nameserver 157.27.10.10
```

Un programma per shell che risolve i nomi in indirizzi IP è `dig` che invoca un server DNS e stampa a video la risposta.

Esempio:

```
$ dig www.univr.it
```

mostra l'IP del server web di Ateneo cioè 157.27.6.235

2 Analisi di alcuni protocolli di livello applicazione

I protocolli di livello applicazione dell'architettura TCP/IP possono imbustare i dati in pacchetti TCP oppure UDP. Le applicazioni che richiedono una trasmissione affidabile (ad es. WEB e posta elettronica) utilizzano TCP che permette di vedere il collegamento tra due host come un “tubo” virtuale in cui i byte sono trasferiti in maniera ordinata ed affidabile. In tale caso i protocolli di livello applicazione dell'architettura TCP/IP sono realizzati mediante scambio di messaggi testuali in modalita' client/server; un programma client (ad esempio il browser Web oppure il programma per leggere la posta) apre una connessione TCP verso il server (Web o di posta) e invia richieste in formato testuale ricevendo le corrispondenti risposte.

Scopo dell’esercizio è sperimentare l’uso del tool `telnet` per interagire con programmi server che utilizzino connessioni di tipo STREAM character oriented.
Il tool `telnet` è un semplice client che apre una connessione TCP con il server e permette all’utente che voglia fare il debug di colloquiare con il server tramite un’interfaccia a caratteri.
Per aprire una connessione con un server in attesa sulla porta N ad un host di nome hostname è sufficiente dare il comando:

```
$ telnet hostname N
```

A questo punto, ammesso che il tentativo di connessione sia andato a buon fine, si interagisce direttamente con il server tramite il videoterminale.

2.1 Il protocollo Hyper Text Transfer Protocol (HTTP)

HTTP è l'acronimo di Hyper Text Transfer Protocol (protocollo di trasferimento di un ipertesto). Usato come principale sistema per la trasmissione di informazioni sul web. Le specifiche del protocollo sono attualmente in carica al W3C (World Wide Web Consortium). La prima versione, la 0.9, dell'HTTP risale alla fine degli anni '80 del XX secolo e costituiva, insieme con l'HTML e gli URL, il nucleo base della "World-Wide Web WWW global information initiative" portata avanti da Tim Berners-Lee al CERN di Ginevra per la condivisione delle informazioni tra la comunità dei fisici delle alte energie. La prima versione effettivamente disponibile del protocollo, la HTTP/1.0, venne implementata dallo stesso Berners-Lee nel 1991 e proposta come RFC 1945 all'ente normatore IETF nel 1996. Con la diffusione di NCSA Mosaic, un browser grafico di facile uso, il WWW conobbe un successo crescente e divennero evidenti alcuni limiti della versione 1.0 del protocollo, in particolare:

- l'impossibilità di ospitare più siti www sullo stesso server (virtual host)
- il mancato riuso delle connessioni disponibili
- l'insufficienza dei meccanismi di sicurezza

Il protocollo venne quindi esteso nella versione HTTP/1.1, presentato come RFC 2068 nel 1997 e successivamente aggiornato nel 1999 come descritto dal RFC 2616

L'HTTP funziona su un meccanismo richiesta/risposta (client/server): il client esegue una richiesta ed il server restituisce la risposta. Nell'uso comune il client corrisponde al browser ed il server al sito web. Vi sono quindi due tipi di messaggi HTTP: messaggi richiesta e messaggi risposta.

HTTP differisce da altri protocolli di livello 7 come FTP, per il fatto che le connessioni vengono generalmente chiuse una volta che una particolare richiesta (o una serie di richieste correlate) è stata soddisfatta. Questo comportamento rende il protocollo HTTP ideale per il World Wide Web, in cui le pagine molto spesso contengono dei collegamenti (link) a pagine ospitate da altri server.

Il messaggio richiesta è composto di tre parti:

- Riga di richiesta (request line)
- Sezione Header (informazioni aggiuntive)
- Body (corpo del messaggio)

La riga di richiesta è composta dal metodo, URI e versione del protocollo. Il metodo di richiesta, per la versione 1.1, può essere uno dei seguenti:

- GET
- POST
- HEAD
- PUT
- DELETE
- TRACE
- OPTIONS

l'URI sta per *Uniform Resource Identifier* ed indica l'oggetto della richiesta (ad esempio la pagina web che si intende ottenere).

I metodi HTTP più comuni sono GET, HEAD e POST. Il metodo GET è usato per ottenere il contenuto della risorsa indicata come URI (come può essere il contenuto di una pagina HTML). HEAD è analogo a GET, ma restituisce solo i campi dell'header, ad esempio per verificare la data di modifica del file. Una richiesta con metodo HEAD non prevede l'uso del body.

Il metodo POST è usato di norma per inviare informazioni al server (ad esempio i dati di un form). In questo caso l'URI indica che cosa si sta inviando e il body ne indica il contenuto.

Gli header di richiesta più comuni sono:

Host: Nome del server a cui si riferisce l'URI. È obbligatorio nelle richieste conformi HTTP/1.1 perché permette l'uso dei virtual host basati sui nomi.

User-Agent: Identificazione del tipo di client: tipo browser, produttore, versione...

Il messaggio di risposta è composto dalle seguenti tre parti:

- Riga di stato (status-line)
- Sezione header
- Body (contenuto della risposta)

La riga di stato riporta un codice a tre cifre catalogato nel seguente modo:

- 1xx: Informational (messaggi informativi)
- 2xx: Success (la richiesta è stata soddisfatta)
- 3xx: Redirection (non c'è risposta immediata, ma la richiesta è sensata e viene detto come ottenere la risposta)
- 4xx: Client error (la richiesta non può essere soddisfatta perché sbagliata)
- 5xx: Server error (la richiesta non può essere soddisfatta per un problema interno del server)

Nel caso più comune il server risponde con un codice 200 (OK) e fornisce il contenuto nella sezione body. Altri casi comuni sono:

- *301 Moved Permanently*. La risorsa che abbiamo richiesto non è raggiungibile perché è stata spostata in modo permanente.
- *302 Found*. La risorsa è raggiungibile con un altro URI indicato nel header Location. Di norma i browser eseguono la richiesta all'URI indicato in modo automatico senza interazione dell'utente.
- *400 Bad Request*. La risorsa richiesta non è comprensibile al server.
- *404 Not Found*. La risorsa richiesta non è stata trovata e non se ne conosce l'ubicazione. Di solito avviene quando l'URI è stato indicato in modo incorretto, oppure è stato rimosso il contenuto dal server.
- *500 Internal Server Error*. Il server non è in grado di rispondere alla richiesta per un suo problema interno.
- *505 HTTP Version Not Supported*. La versione di http non è supportata.

Gli header della risposta più comuni sono:

- *Server*. Indica il tipo e la versione del server. Può essere visto come l'equivalente dell'header di richiesta User-Agent
- *Content-Type*. Indica il tipo di contenuto restituito. La codifica di tali tipi (detti Media type) è registrata presso lo IANA (Internet Assigned Number Authority); essi sono detti tipi MIME (Multimedia Internet Message Extensions), la cui codifica è descritta nel documento RFC 1521. Alcuni tipi usuali di tipi MIME incontrati in una risposta HTML sono:
    - text/html. Documento HTML
    - text/plain. Documento di testo non formattato
    - image/jpeg. Immagine di formato JPEG

Quando un client (ad es. un web browser) richiede una pagina ad un server web, apre la connessione TCP sull'IP del server (eventualmente ottenuto tramite il nome) sulla porta 80. A questo punto scrive nella connessione TCP (esattamente come fosse un file) la riga di richiesta e la sezione header seguita da una riga vuota (carattere di *new_line* ottenibile da tastiera premendo INVIO).

Ad esempio, se si vuole emulare il comportamento di un browser nell'interrogare il server Web di `it.wikipedia.org` occorre scrivere:

```
$ telnet it.wikipedia.org 80
```

```
GET /wiki/Pagina_principale HTTP/1.1
User-Agent: Mozilla/5.0 (compatible; Konqueror/3.2; Linux) (KHTML, like Gecko)
Accept: text/html
Accept-Charset: iso-8859-1, utf-8;q=0.5, *;q=0.5
Accept-Language: en
Host: it.wikipedia.org
[INVIO]
```

A questo punto il server risponde scrivendo nella connessione TCP in caratteri testuali l'header della risposta e la pagina web richiesta (in formato Hyper Text Markup Language – HTML).

Il precedente esempio funziona nelle reti in cui non è obbligatorio l'uso di un proxy web per navigare in Internet. Siccome nei laboratori di Facoltà tutte le richieste web devono passare attraverso il proxy `proxy.sci.univr.it` allora il precedente esempio deve essere trasformato.

Prima occorre collegarsi al proxy della Facoltà

```
$ telnet proxy.sci.univr.it 8080
```

```
CONNECT it.wikipedia.org:80 HTTP/1.1
[INVIO]
```

a questo punto il proxy effettua la connessione per noi e ci manda il messaggio

```
HTTP/1.0 200 Connection established
```

a questo punto è possibile richiedere la pagina nel modo consueto (ma tramite il proxy)

```
GET /wiki/Pagina_principale HTTP/1.1
User-Agent: Mozilla/5.0 (compatible; Konqueror/3.2; Linux) (KHTML, like Gecko)
Accept: text/html
Accept-Charset: iso-8859-1, utf-8;q=0.5, *;q=0.5
Accept-Language: en
Host: it.wikipedia.org
[INVIO]
```

2.2 Il Post Office Protocol (POP)

Il Post Office Protocol (detto anche POP) è un protocollo che ha il compito di permettere, mediante autenticazione, l'accesso ad un account di posta elettronica presente su di un host per scaricare le e-mail del relativo account. Il demone pop (nella versione 3) rimane in attesa sulla porta 110 dell'host (di default, ma può anche essere diversa) per una connessione TCP da parte di un client. I messaggi di posta elettronica, per essere letti, devono essere scaricati sul computer (questa è una notevole differenza rispetto all'IMAP), anche se è possibile lasciarne una copia sull'host. Il protocollo POP3 non prevede alcun tipo di cifratura, quindi le password utilizzate per l'autenticazione fra server e client passano in chiaro. Per risolvere questo possibile problema è stato sviluppata l'estensione APOP che utilizza MD5.

Esercizio: si simuli il comportamento di un client di posta utilizzando il programma telnet e aprendo una connessione verso profs.sci.univr.it sulla porta 110. Non appena il server risponde dare i seguenti comandi attendendo per ognuno la risposta del server.

```
USER master_reti
PASS wer3555sez
LIST
RETR 651
QUIT
```

3 Analisi del traffico di rete

Esistono strumenti SW che consentono di analizzare tutti i pacchetti che arrivano alla propria interfaccia di rete:

- ***tcpdump*** : è un *sniffer* a linea di comando; permette di monitorare il traffico di rete con la possibilità di utilizzare filtri per limitare la quantità di dati, ricavati dall'osservazione della rete secondo varie regole di matching per i pacchetti.
- ***wireshark*** (ex ***ethereal***) : è un software grafico di analisi di protocollo utilizzato per la risoluzione di problemi di rete, per l'analisi e per lo sviluppo di software e di protocolli di comunicazione e per la didattica. Le funzionalità che offre sono molto simili a quelle di tcpdump, ma in più è dotato di un'interfaccia grafica e di più funzionalità di ordinamento e filtraggio.
- ***tethereal*** : è una versione testuale di wireshark.

Al fine di una completa comprensione dei contenuti è prerequisito importante la conoscenza dei fondamenti di rete soprattutto nel contesto TCP/IP.

Per approfondimenti sui comandi e sulle opzioni si faccia riferimento ai manuali dei tool, scaricabili da:

- **tcpdump** http://www.tcpdump.org/tcpdump_man.html
- **ethereal** (**wireshark**) http://www.wireshark.org/docs/
- **thethereal** http://www.ethereal.com/docs/man-pages/tethereal.1.html

Questi tool di analisi si basano tutti sulla libreria C chiamata ***libpcap***. La libreria in questione è nata intorno al 1993 all'Università della California. È supportata pienamente dalla comunità opensource ed è reperibile al sito http://www.tcpdump.org (tcpdump è lo sniffer per eccellenza che sfrutta la libpcap). Le principali funzioni di questa libreria sono la possibilità di cercare e trovare device di rete (intesi come network adapter), gestire potenti filtri di cattura, analizzare pacchetto per pacchetto. Permette la gestione degli errori, quindi un buon livello di debug, ed infine ottimi strumenti per statistiche sulle catture.

3.1 Scaricamento e installazione

I tool si possono scaricare liberamente dalle rispettive pagine web o probabilmente sono presenti già nell'installazione della propria distribuzione Linux.

Esistono tool di analisi di protocolli di rete per l'ambiente Windows, ad esempio ***Analyzer*** del Politecnico di Torino (http://analyzer.polito.it/) oppure ***WinDump*** prelevabile al sito http://www.winpcap.org/windump.

ATTENZIONE: per poter utilizzare le funzionalità di cattura di questi tool in ambiente Linux bisogna essere autenticati come utente *root* o aver installato il tool con *setuid* a root. In ogni caso questi tool possono essere utilizzati per analizzare catture precedentemente effettuate da utente root e salvate su file.

3.2 Alcuni concetti sullo *sniffing*

Sniffing in **reti ethernet non-switched**: In questo tipo di reti ethernet il mezzo trasmissivo è condiviso tramite un *hub* centrale, quindi tutte le schede di rete dei computer nella rete locale ricevono tutti i pacchetti, anche quelli destinati ad altri, selezionando i propri a seconda dell'indirizzo ***MAC*** (indirizzo hardware specifico della scheda di rete). Lo sniffing in questo caso consiste nell'impostare sull'interfaccia di rete la cosiddetta **modalità promiscua**, che disattiva il "filtro hardware" basato sul MAC permettendo al sistema l'ascolto di tutto il traffico passante sul cavo.

Sniffing in **reti ethernet switched**: In questo caso l'apparato centrale della rete, definito switch, si preoccupa, dopo un breve transitorio, di inoltrare su ciascuna porta solo il traffico destinato al dispositivo collegato a quella porta; ciascuna interfaccia di rete riceve, quindi solo i pacchetti destinati al proprio indirizzo, i pacchetti multicast e quelli broadcast. L'impostazione della modalità promiscua è quindi insufficiente per poter intercettare il traffico in una rete gestita da switch. Un metodo per poter ricevere tutto il traffico dallo switch da una porta qualunque è il **MAC flooding**. Tale tecnica consiste nell'inviare ad uno switch pacchetti appositamente costruiti per riempire la *CAM table* dello switch di indirizzi MAC fittizi. Questo attacco costringe lo switch ad entrare in una condizione detta di *fail open* che lo fa comportare come un hub, inviando così gli stessi dati a tutti gli apparati ad esso collegati.

3.3 Utilizzo di *tcpdump*

Questo applicativo è un tool di cattura, attraverso il quale è possibile monitorare il traffico in una rete. Il tool permette, tra l'altro, di limitare la cattura dei pacchetti impostando dei filtri basati, ad esempio, sull'interfaccia di ascolto, sul protocollo o sulla porta utilizzata. Sono inoltre disponibili una serie di opzioni, in particolare vi è la possibilità di limitare il numero di pacchetti catturati o quanti byte acquisire per ciascun pacchetto. Inoltre è possibile salvare su file i pacchetti catturati per leggerli con lo stesso programma in un secondo tempo.

Si riportano alcuni esempi di utilizzo; si noti che per catturare pacchetti occorre lanciare il programma dall'utente root mentre per analizzare file di catture precedenti si può essere utenti non privilegiati.

Utilizzo con le impostazioni di default e senza salvare i pacchetti catturati (vengono solo visualizzati):

```
$ /usr/sbin/tcpdump
```

Analizziamo il comando con le varie opzioni utilizzate:

```
$ /usr/sbin/tcpdump -i eth0 -w eth0.log -s 0 tcp port 80
```

-i eth0 : rappresenta l'interfaccia dalla quale si intende catturare

-w eth0.log : rappresenta il nome del file in cui verranno messi i pacchetti catturati

-s 0 : la flag *-s* permette di specificare quanti byte acquisire da ogni singolo pacchetto (default 68); con *-s0* si richiede di acquisire l'intero pacchetto

tcp : le catture da effettuarsi saranno relative a questo protocollo

port 80 : questo campo limita le catture ai soli pacchetti che utilizzano come source o destination port la porta 80, che, nel caso TCP, è la porta utilizzata dal protocollo *HTTP* (*Hyper Text Transport Protocol*) che realizza il servizio Web.

Per interrompere il monitoraggio occorre utilizzare la combinazione *CTRL + C*.

Altri esempi di utilizzo di tcpdump:

1) viene selezionata l'interfaccia *eth0* e catturato il flusso da e verso *edalab-srv01.sci.univr.it*, scrivendo sul file *eth0.log*

```
$ /usr/sbin/tcpdump -i eth0 -w eth0.log host edalab-srv01.sci.univr.it
```

2) come il precedente esempio ma in questo caso vengono letti solo i pacchetti IP contenenti TCP.

```
$ /usr/sbin/tcpdump -i eth0 -w eth0.log \
  ip proto tcp host edalab-srv01.sci.univr.it
```

3) come il precedente ma in questo caso vengono letti solo i pacchetti da e verso la porta 80 (http).

```
$ /usr/sbin/tcpdump -i eth0 -w eth0.log \
  ip proto tcp host edalab-srv01.sci.univr.it \
  and port 23
```

4) come il precedente esempio ma in questo caso vengono letti solo i pacchetti verso l'host *edalab-srv01.sci.univr.it (dst* sta per *destination* mentre *src* è *source)*

```
$ /usr/sbin/tcpdump -i eth0 -w eth0.log \
  ip proto tcp \
  dst host edalab-srv01.sci.univr.it \
  and port 80
```

5) come il precedente esempio ma in questo caso vengono visualizzati i pacchetti precedentemente salvati nel file eth0.log; anche in questo caso si può applicare un filtro che permette di visualizzare solo i pacchetti di interesse

```
$ /usr/sbin/tcpdump -r eth0.log host edalab-srv01.sci.univr.it
```

Si ricorda che non serve essere root quando TCPDUMP legge un file precedentemente catturato.

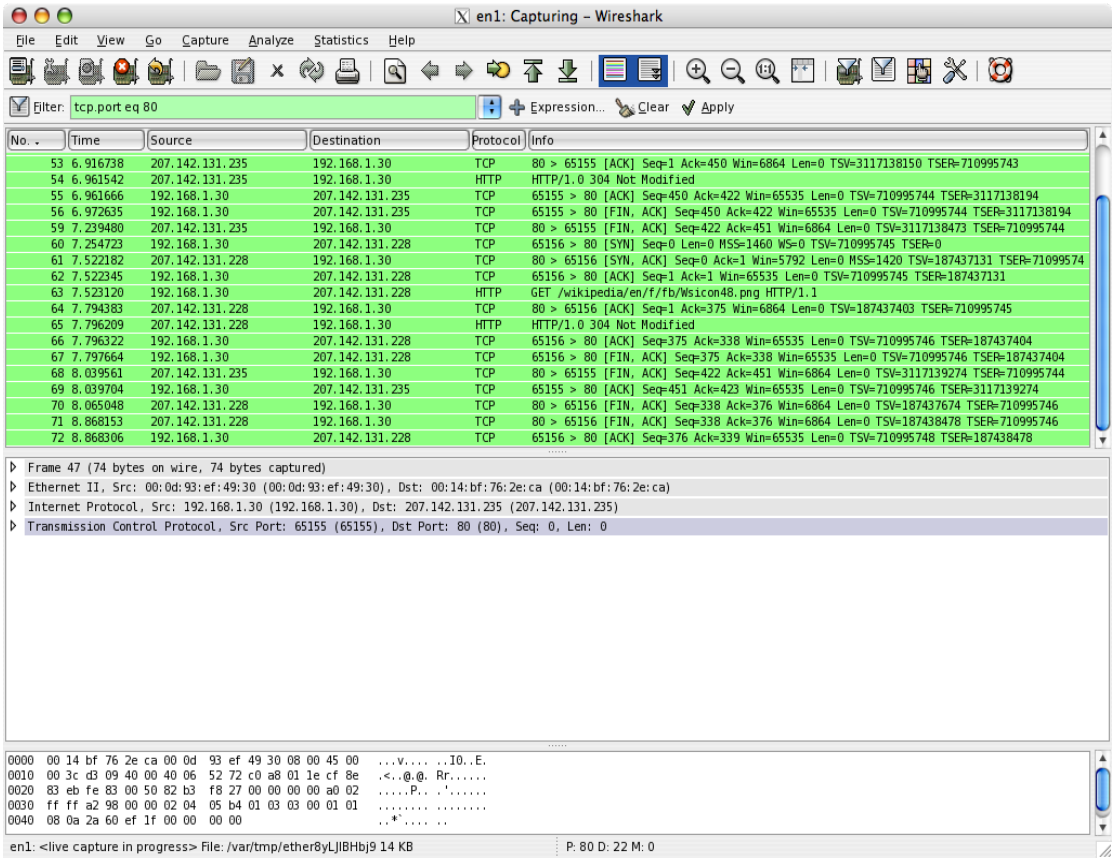
Le capacità di analisi e filtraggio vanno ben oltre questi esempi introduttivi, si consiglia di far riferimento alle pagine del manuale per verificarne le potenzialità.

3.4 Utilizzo di *wireshark*

Per lanciare l'applicazione:

```
$ wireshark
```

si ricordi che è necessario essere l'utente *root* se si intende fare una cattura diretta dalle interfacce di rete.



Alcune caratteristiche:

- i dati possono essere acquisiti “from the wire” (direttamente dal cavo) oppure possono essere letti su un file di cattura precedente
- i dati possono essere catturati dal vivo da reti Ethernet, IEEE 802.11, Token Ring, ecc.
- i dati di rete catturati possono essere esplorati tramite un'interfaccia grafica
- i filtri di visualizzazione possono essere usati per colorare o evidenziare selettivamente le informazioni sommarie sui pacchetti
- la visualizzazione dei dati può essere filtrata
- i protocolli di comunicazione possono essere scomposti, in quanto riesce a “comprendere” la struttura dei diversi protocolli di rete, quindi è in grado di visualizzare incapsulamenti e campi singoli, ed di interpretare il loro significato
- è possibile estrarre il contenuto del livello applicazione di una sessione TCP.

Il modo più immediato per avviare l'ascolto sull'interfaccia di rete di default è dal menu in alto di wireshark `Capture/Start o da Capture/Options`

E' possibile selezionare su quale interfaccia porsi in ascolto con la voce `Capture/Interface`

Per raffinare il processo di analisi e cominciare ad applicare dei filtri usare la voce `Capture/Options/Capture Filter [Start]`
In questo modo a priori si impone uno o più filtri rendendo più piccolo il file di log.

Dopo aver acquisito lo stream di dati è possibile analizzarlo a posteriori sempre mediante filtri con la voce `Analyze/Display Filter`

E' possibile seguire l'intero flusso di dati di una “conversazione” TCP mediante la voce `Analyze/Follow TCP Stream`

Infine è possibile effettuare delle statistiche generali con la voce `Statistics/Summary`

oppure osservare come si ridistribuisce il traffico sulla pila dei protocolli con la voce `Statistics/Protocol Hierarchy`

oppure individuare le conversazioni avvenute tra host con la voce `Statistics/Conversations`

4 Analisi di traffico catturato con TCPDUMP

4.1 Analisi generale del file di cattura

Si può analizzare il contenuto del file `cattura.cap` con TCPDUMP presente su computer di laboratorio oppure con WIRESHARK/ETHEREAL se installato sul proprio computer. Con TCPDUMP occorre usare il seguente comando (si può redirigere l'output su un file di testo da aprire con il proprio editor preferito).

```
$ /usr/sbin/tcpdump -e -vvv -r cattura.cap > out.txt
$ less out.txt
```

L'output mostra una descrizione di ogni pacchetto catturato. Ogni descrizione contiene (si veda come esempio il primo pacchetto):

- l'istante di cattura (es. 09:22:23.490424)

- header di livello 2
  - MAC sorgente (es. 00:11:43:a7:19:fa)
  - MAC destinazione (es. bradcast cioè ff:ff:ff:ff:ff:ff)
  - Protocol Type (es. IPv4 cioè 0x0800)
- header di livello 3
  - campi vari (es. tos=0x00, ttl=64, id=60927, offset=0, flags [none], proto: UDP (17), length=223)
  - sorgente IP (es. 157.27.242.154)
  - destinazione IP (es. 157.27.242.255)
- header di livello 4
  - porte sorgente e destinazione accorpate per comodità ai corrispondenti IP (es. netbios-dgm)
  - verifica della checksum
- header di livello applicazione (es. Netbios, protocollo di condivisione di dischi in rete).

Il file `cattura.cap` contiene pacchetti catturati da `tcpdump` effettuata dall'interfaccia di rete avente MAC 00:11:43:3E:98:DB e IP 157.27.242.239. Analizzando in maniera sommaria l'output si nota che sono stati catturati anche pacchetti la cui sorgente o destinazione non coincide con tali indirizzi; questi pacchetti tuttavia sono passati sul doppino di rete su cui ascoltava TCPDUMP e quindi sono stati catturati. Alcuni di questi pacchetti pacchetti hanno destinazione MAC o IP di tipo multicast o broadcast e quindi è naturale che non siano stati filtrati dallo switch. Altri pacchetti unicast sono invece stati inoltrati su tutte le porte dello switch perché esso non conosceva l'esatta porta di uscita.

Analizzando i pacchetti uno per uno si possono scoprire cose molto interessanti sui servizi e sulle macchine presenti in quel momento in rete...

4.2 Analisi di una connessione al server di posta

Il file `cattura.cap` contiene pacchetti catturati da `tcpdump` effettuata dall'interfaccia di rete avente MAC 00:11:43:3E:98:DB e IP 157.27.242.239. Mentre `tcpdump` stava catturando è stato svolto l'esercizio della Sezione 2.2 (accesso al server di posta POP). Come si può notare, il file contiene un sacco di pacchetti che spesso non fanno riferimento all'host da cui si è effettuata la lettura della posta.

Un modo per diminuire il volume di pacchetti visualizzati è impostare un filtro di lettura sul MAC del mio host nel modo seguente

```
$ /usr/sbin/tcpdump -e -vvv -r cattura.cap ether host 00:11:43:3E:98:DB >
mymac.txt
```

in questo modo è possibile notare uno scambio preliminare di messaggi con `proxy.sci.univr.it` che il server DNS.

Se invece voglio isolare solo i pacchetti scambiati con il server di posta (`profs.sci.univr.it`) allora occorre usare un filtro con l'IP di tale host (si può ricavare con il comando `dig`)

```
$ /usr/sbin/tcpdump -e -vvv -r cattura.cap ip host 157.27.252.10 > myhost.txt
```

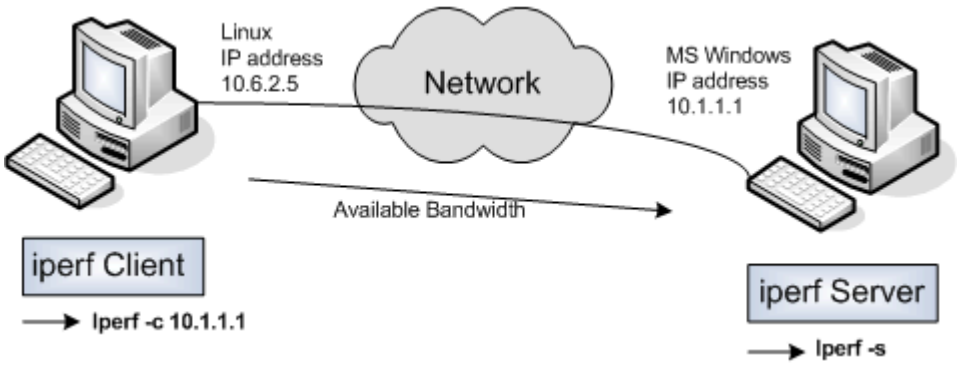
con il comando

```
$ /usr/sbin/tcpdump -X -r cattura.cap ip host 157.27.252.10 > myhost.txt
```

è possibile esaminare anche il contenuto dei pacchetti (si noti la password dell'account di posta ben visibile a occhi indiscreti).

5 Iperf

Iperf è un tool sviluppato da National Laboratory for Applied Network Research (NLANR) e da Distributed Applications Support Team (DAST).
Iperf misura la capacità di una rete e aggiuntivamente riporta jitter e la perdita di pacchetti datagram.



Iperf è un'applicazione client/server e pertanto deve essere lanciato su un host come server e su un altro come client. Esiste un unico eseguibile, che si comporta come client o server a seconda dei parametri passati.

Ad esempio:

```
$ server: iperf -s
$ client: iperf -c server
```

crea un server (-s) ed un client che ad esso si connetta (-c e poi l'IP del server).

Un possibile esempio di output è il seguente:

```
[  3] local 127.0.0.1 port 5001 connected with 127.0.0.1 port 3877
[ ID] Interval       Transfer     Bandwidth
[  3] 0.0-10.0 sec   596 MBytes   500 Mbits/sec
```

dove l'ID è utilizzato per identificare le connessioni poiché iperf server supporta connessioni multiple.

Come default, iperf usa una connessione TCP, ed il client si connette tramite la porta 5001.

Nella tabella è riassunto brevemente il significato di alcuni possibili parametri.

Parametri	Significato
-s	esegue in modalità server
-c	esegue in modalità cliente (specificando il server a cui connettersi)
-u	utilizza il protocollo udp
-b	ampiezza di banda da utilizzare (in bps)
-t	durata della trasmissione (in secondi)
-i	intervallo di segnalazione delle statistiche (in secondi)

6 Esercizi

Esercizio 1: lanciare ping indicando rispettivamente l'indirizzo di loopback, il proprio IP e il default gateway e verificare il round trip time nei tre casi. Cambia nel tempo anche per lo stesso IP ? Qual è il minore ?

Esercizio 2: lanciare ping indicando indirizzi IP della propria rete (157.27.241.X) e verificando se sono raggiungibili.

Esercizio 3: visualizzare il contenuto della cache prima e dopo aver lanciato il comando ping su un host della stessa rete nell'esercizio precedente e verificare cosa cambia. Visualizzare la cache dopo 15 minuti e verificare cosa cambia.

Esercizio 4: si simuli il comportamento di un client di posta utilizzando il programma telnet e aprendo una connessione verso profs.sci.univr.it sulla porta 110. Non appena il server risponde dare i seguenti comandi attendendo per ognuno la risposta del server.

```
USER master_reti
PASS wer3555sez
LIST
RETR 651
QUIT
```

Esercizio 5: utilizzare tcpdump o wireshark in modo da catturare sull'interfaccia principale di ricezione del traffico della macchina, tutti i pacchetti destinati al proprio indirizzo (niente modalità promiscua). Quindi "pingare" l'indirizzo della macchina (dove sti sta eseguendo tcpdump) e controllare il traffico ricevuto.

Esercizio 6: Usare tcpdump o wireshark in modo da catturare su file i primi 100 pacchetti provenienti o destinati a google.com sulla porta http.

Esercizio 7: Usare wireshark per "sniffare" la password dell'utente *master_reti* che si connette all'host *profs.sci.univr.it* tramite *TELNET* sulla porta 110 per leggere la posta come nel Cap. 2.2.

Esercizio 8: Usare iperf per misurare la capacità della rete di laboratorio. Testare usando le seguenti modalità:

- TCP
- UDP
- UDP bidirezionale
- UDP mentre è in esecuzione anche tcpdump